

Motion planning and bimanual coordination in humanoid robots¹

Pietro MORASSO^{a,b,2}, Vishwanathan MOHAN^a,
Giorgio METTA^{a,b}, Giulio SANDINI^{a,b}

^a*Italian Institute of Technology, Genoa, Italy*

^b*Department of Informatics, Systems, Telematics, University of Genoa, Italy*

Abstract. Humanoid robots have a large number of “extra” joints, organized in a humanlike fashion with several kinematic chains. In this chapter we describe a method of motion planning that is based on an artificial potential field approach (Passive Motion Paradigm) combined with terminal-attractor dynamics. No matrix inversion is necessary and the computational mechanism does not crash near kinematic singularities or when the robot is asked to achieve a final pose that is outside its intrinsic workspace: what happens, in this case, is the *gentle degradation* of performance that characterizes humans in the same situations. Moreover, the remaining error at equilibrium is a valuable information for triggering a reasoning process and the search of an alternative plan. The terminal attractor dynamics implicitly endows the generated trajectory with human-like smoothness and this computational framework is characterized by a feature that is crucial for complex motion patterns in humanoid robots, such as bimanual coordination or interference avoidance: precise control of the reaching time.

Keywords: Motion planning, Humanoid robot, Attractor dynamics, Terminal attractors.

Introduction

Humanoid robots have a large number of “extra” joints, organized in a humanlike fashion with several kinematic chains. Consider, for example, Cog [1] with 22 DoFs (Degrees of Freedom), DB [2] with 30 DoFs, Asimo [3] with 34 DoFs, H7 [4] with 30 DoFs, iCub [5] with 53 DoFs. When a finger touches a target, the elbow might be up or down and the trunk may be bent forward, backward or sideways. Thus an infinite number of solutions are available to the motor planner/controller. This redundancy is advantageous because it enables a robot to avoid obstacles and joint limits and attain more desirable postures, for example when it is not sufficient to simply tap a target because a precise force vector must be applied to the touched object. From a control and learning point of view, however, redundancy also makes it quite complicated to

¹ IOS press, series KBIES (Knowledge-Based Intelligent Engineering Systems); subseries of "Frontiers in Artificial Intelligence and Applications"; book title "Computational Intelligence and Bioengineering"; editors: F. Masulli, A. Micheli, A. Sperduti (2009).

² Corresponding author: Pietro Morasso, University of Genoa, DIST, Via Opera Pia 13, 16145 Genoa, Italy. Email: pietro.morasso@unige.it

find good movement plans that do not crash when it turns out that the designated target is unreachable or barely reachable. How do humans decide what to do with their extra joints, and how should humanoid robots control all their joints in order to generate coordinated movement patterns? Moreover, is the selection/coordination of redundant DoFs independent of the spatio-temporal organization of the reaching movements? Early studies of human arm trajectory formation [6,7] showed invariant spatio-temporal features, such as a symmetric bell-shaped speed profile, which can be explained in terms of minimization of some measure of *smoothness*, such as jerk [8] or torque-change [9]. Later studies emphasized the importance of physical or computational force fields in the neural control of movement or motor learning [10,11,12]. Most approaches to motion planning in robotics were derived from the early study of Whitney [13] named RMRC (resolved motion rate control), which is based on the real-time inversion of the Jacobian matrix of the kinematic transformation, i.e. the function that links the joint angle vector q of a kinematic chain to the pose x of the end-effector. Clearly, for redundant kinematic chains RMRC must be modified by using one form or another of pseudo-inversion, as the Moore-Penrose matrix that provides a minimum norm solution for dq or other more general pseudo-inversion methods [14] that can associate an arbitrary cost function to the inversion calculation. Another method (Extended Jacobian Method [15,16]) extends the usual Jacobian matrix with additional rows that take into account virtual movements in the null space of the kinematic transformation: the extended Jacobian matrix is square and can be inverted in the usual way. In any case, the classical approaches to robot planning/control work well only inside the workspace and far away from kinematic singularities. If this is acceptable for an industrial robot, which has no or a limited number of excess DoFs and operates in a well defined and predictable environment, it is not feasible for a humanoid robot supposed to carry out, as humans, activities of daily life in a generally unknown environment.

In this chapter we further develop for the humanoid robot scenario a method of motion planning that is based on an artificial potential field approach (PMP: Passive Motion Paradigm [17]) combined with terminal-attractor dynamics [18] that has also been applied to robot reasoning [19]. PMP is an example of a class of robot planning methods which are based on non-linear, attractor dynamics. In particular, PMP is somehow similar to the VITE model (Vector Integration To Endpoint [20]). In both cases there is a “difference vector” associated with an attractor dynamics that has a point attractor in the designated target. The difference is that VITE focuses on the neural signals commanding a pair of agonist-antagonist muscles, whereas the PMP model focuses, at the same time, on the trajectories in the extrinsic and intrinsic spaces. In comparison with a recent paper by Hersch and Billard [21] that builds upon the VITE model, the extension of the PMP model described in this chapter is equally well a “multi-referential dynamical systems” for implementing reaching movements in complex, humanoid robots but does not require any explicit inversion and/or optimisation procedure. Another approach to motion planning, based on non-linear dynamics, builds upon the basic idea of learning attractor landscapes in phase space for canonical dynamical systems with well defined point attractor properties [22]. The approach is very effective for movement imitation, because it approximates the attractor landscape by means of a piecewise-linear regression technique but somehow misses the multi-referential feature mentioned above. Also in the PMP model there is a well defined attractor landscape but it is derived from the composition of different virtual force fields that have a clear cognitive meaning and thus allow the seamless integration of planning with reasoning. Moreover, no matrix inversion is necessary and

the computational mechanism does not crash near kinematic singularities or when the robot is asked to achieve a final pose that is outside its intrinsic workspace: what happens, in this case, is the *gentle degradation* of performance that characterizes humans in the same situations.

2. The computational model.

2.1 *The Passive Motion Paradigm*

The idea behind the Passive Motion Paradigm is that motor commands for any kind of motor action, for any configuration of limbs and for any degree of redundancy can be obtained by an “internal simulation” of a “passive motion” induced by a “virtual force field” [10] applied to a small number of task-relevant parts of the body. Here “internal simulation” identifies the relaxation to equilibrium of an internal model of limb (arm, leg etc, according to the specific paradigm); “passive motion” means that the joint rotation patterns are not specifically computed in order to accomplish a goal but are the indirect consequence of the interaction between the internal model of the limb and the force field generated by the target, i.e. the intended/attended goal. In a general sense, PMP is analogous to the mechanism of coordinating the motion of a wooden marionette by means of attached strings (Fig. 1).

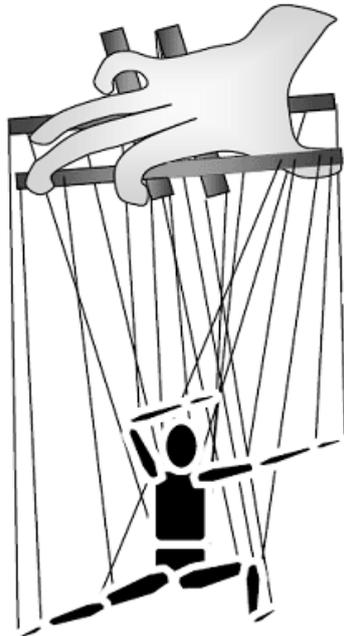


Figure 1. The “marionette” metaphor of the Passive Motion Paradigm (PMP). The “internal model” that coordinates and plans the motion of all the joints operates on a small set of force fields applied to “focal points” of the body model.

What makes it attractive from the computational point of view is its simplicity and robustness. It is simple because the planner/controller does not have to be concerned

with all the DoFs at the same time but only with a smaller number of “end-effectors”. It is robust because no model inversion is necessary and the relaxation to equilibrium in a conservative force field can never crash. It does not require any cost function to be specified in order to solve the indeterminacy related to the excess DoFs but it allows to integrate internal and external constraints (in the intrinsic and extrinsic spaces, respectively). The internal model, associated with the force-field, in a way represents the set of all geometrically possible solutions out of which one is implicitly selected as a function of the structural and task-specific constraints.

2.2 General formulation of the computational model

Let x be the vector that identifies the pose of the end-effector of a robot in the extrinsic workspace and q the vector that identifies the configuration of the robot in the intrinsic joint space: $x = f(q)$ is the kinematic transformation that can be expressed, for each time instant, as follows:

$$\dot{x} = J(q) \cdot \dot{q} \quad (1)$$

where $J(q)$ is the Jacobian matrix of the transformation. The motor planner/controller, which expresses in computational terms the PMP, is defined by the following steps (see fig. 2):

- 1) Associate to the designated target x_T a conservative, attractive force field in the extrinsic space

$$F = K_{ext}(x_T - x) \quad (2)$$

where K_{ext} is the virtual impedance matrix in the extrinsic space. The intensity of this force decreases monotonically as the end-effector approaches the target.

- 2) Map the force field into an equivalent torque field in the intrinsic space, according to the principle of virtual works:

$$T = J^T F \quad (3)$$

- 3) Relax the arm configuration in the applied field:

$$\dot{q} = A_{int} \cdot T \quad (4)$$

where A_{int} is the virtual admittance matrix in the intrinsic space: its modulation does not affect the trajectory of the end-effector but modifies the relative contributions of the different joints to the reaching movement.

- 4) Map the arm movement into the extrinsic workspace:

$$\dot{x} = J \cdot \dot{q} \quad (5)$$

Integrating equation 5 over time we obtain a trajectory in the extrinsic space, whose final position corresponds to an equilibrium configuration x_F . By definition, the trajectory of the end-effector is the unique flowline in the force field passing through

$x(t_0)$ and converging to x_F . The algorithm always converges to a “reasonable” equilibrium state, whatever the degree of redundancy of the robot: if the target is within the workspace of the robot, it is reached ($x_F = x_T$); if it is not reachable, the robot settles on the point of the boundary of the workspace that is at a minimum distance from the target x_T , typically with a full extension of the arm toward the target. The relative values of the elements of the joint admittance matrix A_{int} measure the degree of involvement of each DoF in a given reaching movement. For example, a joint rotation can be “frozen” by setting to zero the corresponding admittance value; in any case, the distribution of admittance values can be task-dependent thus allowing to exploit redundancy in a goal-oriented way. The scheme of figure 2 also includes three additional elements: 1) a force field in the intrinsic space for implementing internal constraints, 2) a force field in the extrinsic space for implementing external constraints, 3) a time varying gain for implementing terminal attractor dynamics.

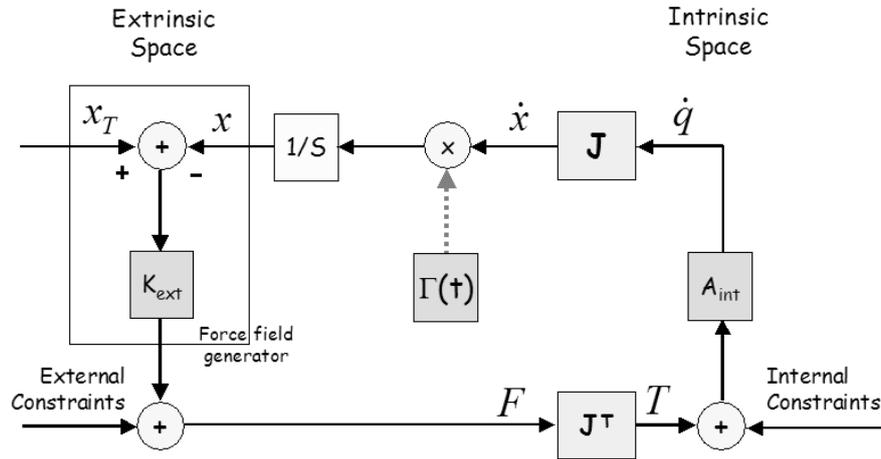


Figure 2. Basic computational scheme of the PMP for a simple kinematic chain. x is the position/orientation of the end-effector, expressed in the extrinsic space; x_T is the corresponding target; q is the vector of joint angles in the intrinsic space; J is the Jacobian matrix of the kinematic transformation $x = f(q)$; K_{ext} is a virtual stiffness that determines the shape of the attractive force field to the target; “external constraints” are force fields in the extrinsic space; “internal constraints” are force fields in the intrinsic space; A_{int} is a virtual admittance that distributes the relaxation motion to equilibrium to the different joints; $\Gamma(t)$ is the time-varying gain that implements the terminal attractor dynamics.

2.2.1 Internal constraints

Typical internal constraints are joint limits: $\{q_i^{\min} < q_i < q_i^{\max}, i = 1, n\}$. They can be implemented by means of an elastic force field in the joint space, with a nominal equilibrium in the middle of the range of motion of each joint. An alternative implementation is a repulsive field that steeply grows in the vicinity of the joint limits. In both cases, this force field in the intrinsic space is added to the force field that attracts the end-effector to the target. The latter force field allows the target to be reached, whereas the former field induces motions in the null space of the kinematic

transformation that select joint configuration patterns compatible with the target but as far as possible from the joint limits.

2.2.2 External constraints

A typical external constraint is an obstacle, which can be implemented as a repulsive force field in the extrinsic space, to be added to the attractive force field to the target. Another problem that can be addressed in a similar way is choosing the “best” configuration, after reaching a target, for delivering a desired force vector at the end-effector. The notion of optimality, in this case, can be related to the torque limits that characterize each actuator: an optimal arm configuration, from the point of view of the actuators, corresponds to a required torque output for each actuator that is as far as possible from the torque limit. This involves a kind of search in the null space of the kinematic transformation because, for a given force vector delivered at the end-effector, the actuator torques depend on the arm configuration via the transpose Jacobian. The solution is given by a simulation of the model, after x has converged to x_T , with the target force F_T applied as an additional input in the extrinsic space. If the virtual stiffness is sufficiently high, such target force will determine a motion in the null space without displacing too much the position of the end-effector. The solution found in this way can be induced to comply with the torque limits by introducing saturation elements of the different actuator torques just after the transpose Jacobian. In general, a basic feature of the proposed model is that it uses the Jacobian and transpose Jacobian matrices for mapping motions and efforts between the intrinsic and extrinsic spaces but in no way requires model inversion, thus eliminating completely the danger of crashing. Redundancy is taken into account not in terms of optimisation of a cost function but in terms of task-dependent mixtures of multiple constraints, expressed as additive force fields in both operational spaces.

2.3 Terminal attractor dynamics

The basic PMP model is an asymptotically stable dynamical system with a point attractor that brings the end-effector to the target if the target is indeed reachable. However, asymptotic stability implies that the equilibrium configuration is reached after an infinite time and does not provide any mechanism to control the speed of approach to equilibrium. A way to explicitly control the time, without using a clock, is to insert in the non-linear dynamics of the PMP model a suitable time-varying gain $\Gamma(t)$ that grows monotonically as x approaches the equilibrium state and diverges to an infinite value in that state. The technique was originally proposed by Zak [23] for speeding up the access to addressable memory in neural networks and then was applied to a number of problems in neural networks. Our purpose, however, is not merely to speed up the operation time of the planner but to allow a control of the reaching time as well as the speed profile in order to fit the human reaching patterns and to allow synchronization of complex tasks as in bimanual coordination. In particular, we propose the following time-varying gain:

$$\begin{cases} \Gamma(t) = \frac{\dot{\xi}}{1-\xi} \\ \xi(t) = 6 \cdot (t/\tau)^5 - 15(t/\tau)^4 + 10(t/\tau)^3 \end{cases} \quad (6)$$

where $\xi(t)$ is a time-base generator (TBG): a scalar function that smoothly evolves from 0 to 1 with a prescribed duration τ and a symmetric bell-shaped speed profile. A simple choice for the TBG is the minimum jerk polynomial function of equation 6, but other types of TBGs are also applicable without any loss of generality. In summary, an extension of the basic PMP model in order to allow terminal attractor dynamics simply requires that equation 4 is substituted by the following one:

$$\dot{x} = -\Gamma(t) \cdot J \cdot \dot{q} \quad (4a)$$

In order to demonstrate that in this way the target is reached after a time equal to τ and with an approximately bell-shaped speed profile, we can substitute the vector equation 4a with an equivalent scalar equation in the variable z defined as the running distance from the target along the trajectory generated by the PMP network ($z = 0$ for $x = x_T$): $\dot{z} = \Gamma(t) f(z)$, where $f(z)$ is, by construction, a monotonically increasing function of z which passes through the origin because $x = x_T$ is the point attractor of the dynamical PMP model. Therefore, for $f(z)$ we can formulate the following linear bound $\gamma_{\min} z < f(z) < \gamma_{\max} z$, where $\gamma_{\min}, \gamma_{\max}$ are two positive constants. By denoting with γ any value inside the $\gamma_{\min} \rightarrow \gamma_{\max}$ interval, we can write the following equation:

$$\frac{dz}{dt} = -\frac{d\xi/dt}{1-\xi} \gamma z \quad (7)$$

from which we can eliminate time

$$\frac{dz}{d\xi} = -\frac{\gamma z}{1-\xi} \quad (8)$$

The solution of this equation is then given by:

$$z(t) = z_0 (1-\xi)^\gamma \quad (9)$$

where z_0 is the initial distance from the target along the trajectory. This means that, as the TBG variable $\xi(t)$ approaches 1, the distance of the end-effector from the target goes down to 0, i.e. the end-effector reaches the target exactly at time $t = \tau$ after movement initiation. Since this applies to both limits of the bound we can write the following bound:

$$z_0 (1-\xi(t))^{\gamma_{\min}} < z(t) < z_0 (1-\xi(t))^{\gamma_{\max}} \quad (10)$$

In any case the terminal attractor $z = 0$ is reached at $t = \tau$. The speed profile may be somehow distorted in relation with a symmetric bell shape (fig. 3 right panel) but the terminal attractor property of the model is maintained for a wide range of values of γ .

2.4 Distributed representation of the Jacobian or transpose Jacobian matrix of a kinematic chain

If a precise analytic expression of the kinematic transformation $x = f(q)$ is available, then the Jacobian can be computed in a direct way: $J = \partial f / \partial q$. In many cases, however, this is not possible or at least the expression of the kinematic transformation is not reliable due to large uncertainties in the geometric parameters of the robot. Still we can obtain experimentally a “training set” of joint rotation readings with the corresponding coordinates of the end-effector.

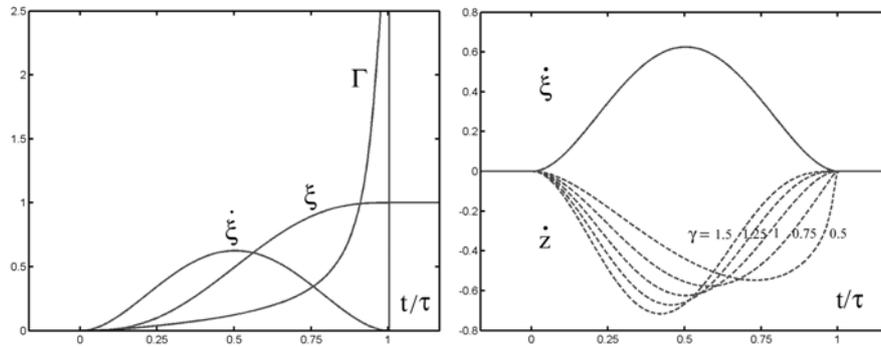


Figure 3. Time-base generator for terminal attractor dynamics - $\Gamma(t)$ - obtained from a minimum jerk time function - $\xi(t)$ - with assigned duration τ .

In this case, we can recover J from samples of the kinematic transformation. This purpose can be achieved by training a standard multi-layer neural network (ANN) with the back-propagation technique and the training set mentioned above. For example, we can use a three-layer network, where $\{q_i\}$ is the input array (joint angles), $\{x_k\}$ is the output array (position/orientation of the end-effector), and $\{z_j\}$ is the output of the hidden units. The following are the equations of the ANN:

$$x = f(q) \Rightarrow \begin{cases} h_j = \sum_i w_{ij} q_i \\ z_j = g(h_j) \\ x_k = \sum_j w_{jk} z_j \end{cases} \quad (11)$$

where $\{w_{ij}\}$ are the connection weights from the input to the hidden layer, $\{w_{jk}\}$ are the connection weights from the hidden to the output layer, $\{h_j\}$ are the net inputs to the neurons of the hidden layer. The neurons of the hidden layer are characterized by the logistic transfer function $g(h)$; the output layer is composed of linear neurons. After training, we can extract the Jacobian matrix from the neural network by applying the chain rule in the following way:

$$J_{ki} = \frac{\partial x_k}{\partial q_i} = \sum_j \frac{\partial x_k}{\partial z_j} \frac{\partial z_j}{\partial h_j} \frac{\partial h_j}{\partial q_i} = \sum_j w_{jk} g'(h_j) w_{ij} \quad (12)$$

Eq. 18 can be easily adapted to ANNs with more than 3 layers. At run-time, the ANN must be fed with the flow of $\{q_i(t)\}$ values in order to recover the corresponding $\{h_j(t)\}$ values. Thus it is possible to carry out the computations indicated in fig. 2 by means of the two following equations:

$$\dot{x} = J \cdot \dot{q} \Rightarrow \dot{x}_k = \sum_i J_{ki} \dot{q}_i = \sum_i \left(\sum_j w_{jk} \cdot w_{ij} \cdot g'(h_j) \right) \dot{q}_i \quad (13)$$

$$T = J^T \cdot F \Rightarrow T_i = \sum_k J_{ik} F_k = \sum_k \left(\sum_j w_{ji} \cdot w_{kj} \cdot g'(h_j) \right) F_k \quad (14)$$

3. The ICUB platform

The iCub is a small humanoid robot of the dimensions of a three and half year old child (figure 4) and designed by the RobotCub consortium, a joint collaborative effort of 11 teams from Europe, three teams from Japan and two teams from USA. The 105 cm tall baby humanoid body is characterized by 53 degrees of freedom: 7 DoFs for each arm, 9 for each hand, 6 for the head, 3 DoFs for the torso and spine and 6 DoFs for each leg. The current design uses 23 brushless motors in the arms, legs, and the waist joints. The remaining 30 DoFs are controlled by Faulhaber DC motors. The iCub body is also endowed with a range of sensors for sensing forces, torques, joint angles, inertial sensors, tactile sensors, 3 axis gyroscopes, cameras and microphones for visual and auditory information acquisition. Most of the joints are tendon driven, some are direct, according to the placement of the actuators which is sort of constrained by the shape of the body.

Apart from the interface API that speaks directly to the hardware, the middle ware of iCub software architecture is based on YARP [24], an open-source software platform that supports distributed computation with a specific impetus given to robot control and efficiency. The main goal of YARP is to minimize the effort devoted to infrastructure-level software development by facilitating modularity, support for simultaneous inter-process communication, image processing, as well as a class hierarchy to ease code reuse across different hardware platforms and hence maximize research-level development and collaboration. With special focus being given on manipulation and interaction of the robot with the real world, iCub is characterized by highly sophisticated hands, flexible oculomotor system and reasonable bimanual workspace.

Intelligent, real time exploitation of the DoFs afforded by complex robotic bodies is the target problem discussed in this article. In contrast to computer simulations of planar arms with few degrees of freedom, using a complex humanoid is also a test of the scalability and real time operation of the dynamical system proposed in this paper. Further, this also necessitates solving a range of engineering (and software) problems in addition to conceptual/scientific ones to facilitate successful planning and execution of complex tasks where many real systems have to simultaneously interact and pass information.

The simulations shown in the next section were carried out by linking various PMP networks to the kinematic/dynamic simulator of the iCub robot [25]. This simulation study is preliminary to the integration of the PMP model in the real-time control architecture. It is also a useful validation tool, in order to verify that the planned kinematic patterns are compatible with the requirements of the actuators in terms of torque, speed, acceleration, etc. In this study, the computational model was used to coordinate all degrees of freedom involved in the left arm-torso-right arm chain of the baby humanoid (i.e. 7+3+7 DoFs in total). Specifically, for each arm we deal with the following joints: shoulder pitch (front-back movement when the arm is aligned with gravity), shoulder roll (adduction, abduction movement of the arm), shoulder yaw (yaw movement when the arm principal axis is aligned with gravity), elbow flexion/extension, wrist prono/supination (rotation along arm principal axis), wrist pitch, wrist yaw.

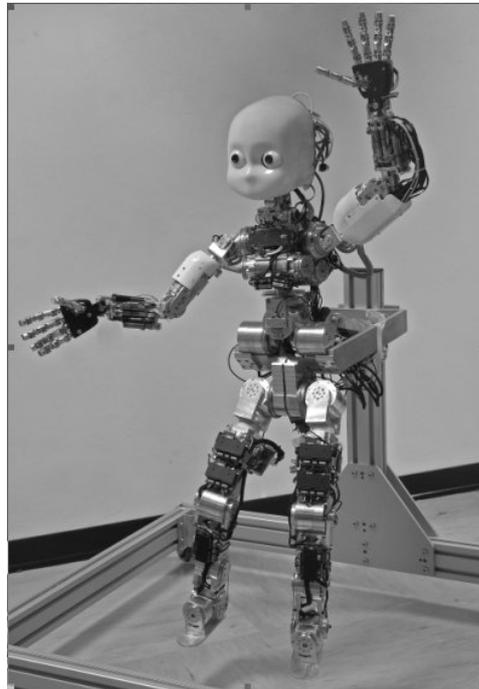


Figure 4. The 53 DoFs iCub robot.

While theoretically six DoFs would already allow reaching any point in the workspace with every attainable orientation, in practice, the seventh DoF provides a means to reach while avoiding interference with vision. This additional flexibility is very much desired if we have to deal with grasping and the interaction with objects in front of the robot while maintaining sight of the action. It is also worth mentioning that the full range of motion for the shoulder can only be obtained by a double joint mechanism similar to the human clavicle and collar bones. The torso is characterized by three DoFs: torso yaw (with respect to gravity), torso roll (lateral movement) and torso pitch (front back movement).

4. Application of the computational architecture to the ICUB platform

The basic scheme of figure 2 can be extended in many ways. For example, consider the following paradigms:

1. Apply two or more force fields to the same kinematic chain, such as the arm, in order to determine the total pose of the hand.
2. Implement a bimanual coordination task, in order to reach at the same time two target points of an object in order to grasp it.

In the former case, there are two force fields: one that pulls the end-effector to the target and the other that attracts the wrist or hand to a proper pose with respect to the target. Irrespective of the redundancy of the system, it is always possible to map the entire distal space to the proximal space by traversing through the horizontal (geometric) and vertical (elastic) links in the computational chain. The joint space hence becomes a natural site where multiple articulatory constraints are concurrently combined, to derive an incremental change in configuration of the body (and motor commands) and this process keeps circulating in the loop till the system achieves an equilibrium configuration.

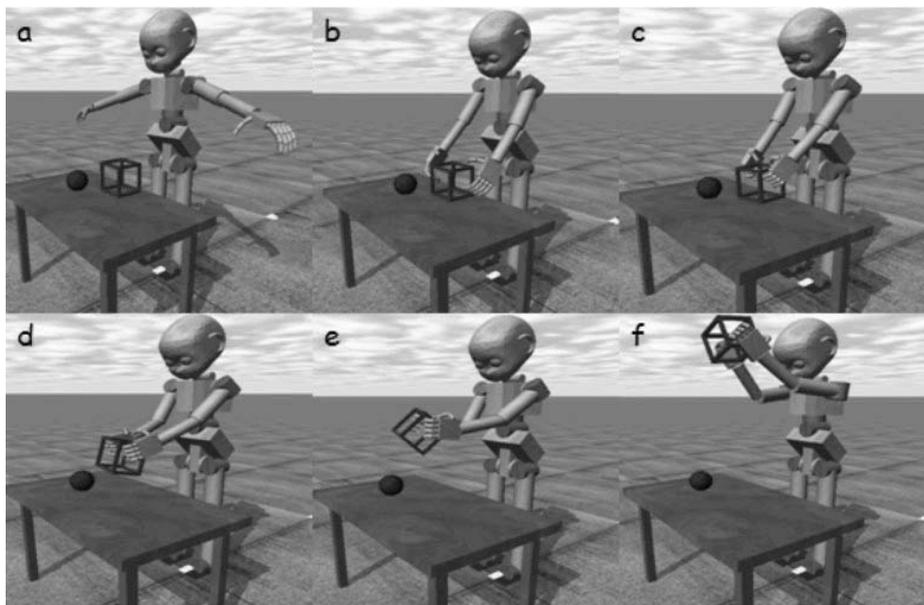


Figure 5. combined task that consists of reaching (frames a and b), grasping (frame c), and then lifting an object (frames d-f).

Let us now consider a combined task that consists of reaching, grasping, and then lifting an object. In spite of the fact that humans (almost unconsciously) execute such tasks with noticeable ease, it is worth noting that in this example even specifying the goal of the task (in computational sense) is far from trivial, if we want an artificial agent to do the same. Once the object is grasped successfully using the two arms, the task of transporting the external object poses stringent constraints both on the movement trajectory of the two arms as well as the timing of of arms motion: both arms must move to the target in such a way that they are always in contact with the

object and any unpredictable effect on the object must be compensated by suitable reconfiguration of the body. Instead of controlling the body that in turn controls the external object and carries it to the goal (which may be extremely complicated to achieve in computational terms considering the number of constraints that must be explicitly accommodated into the controller in order to achieve this successfully), the computational scheme suggested by this paper is to reason in the reverse direction: from the goal to the external object, and then to the body. In this way, the external object in a sense is always kinematically and dynamically coupled with the body. In other words, the external object (e.g. the cube of figure 5) is pulled towards the goal target by one virtual force field; this pull in turn disturbs the end-effectors (both hands) that passively comply to the externally imposed motion; this disturbance then propagates to the proximal space through the Jacobian matrices to derive an incremental change in the joint angles and so on.. Figure 5 shows an example of this combined task: reaching (frames a and b of the top panel), grasping (frame c), and then lifting the object (frames d-f). Figure 6 shows the PMP network that coordinates the lifting phase of the task, with a force field applied to the object and propagated to the PMP sub-networks that correspond to the right arm, left arm, and trunk. Please note that the same time base generator $\Gamma(t)$ coordinates the motion of all the joints of the arm and the wrist.

If the motor commands derived by this process are fed to the robot, the robot will reproduce the same motion. Otherwise, the same simulation can carry out the function of a “mental simulation” of the same task. It is worth noting that the PMP network that carries out the different phases of the task (reaching, grasping, lifting) is dynamically reconfigured. The external object can be a simple cube, as in fig. 12, or a more complicated tool with several new controllable degrees of freedom: for example, two arms linked in parallel to a steering wheel while driving a car. The computational model by itself makes no difference between the representational schema of the motor spaces of the body and the external object. The tool space is represented exactly in the same way as the body, by means of a generalized force and position node, linked vertically by a virtual admittance matrix A_E (characterizing the incremental transformation from force information to position information in the tool space), and horizontally by the device Jacobian matrix J_E that forms the interface between the body and the tool device.

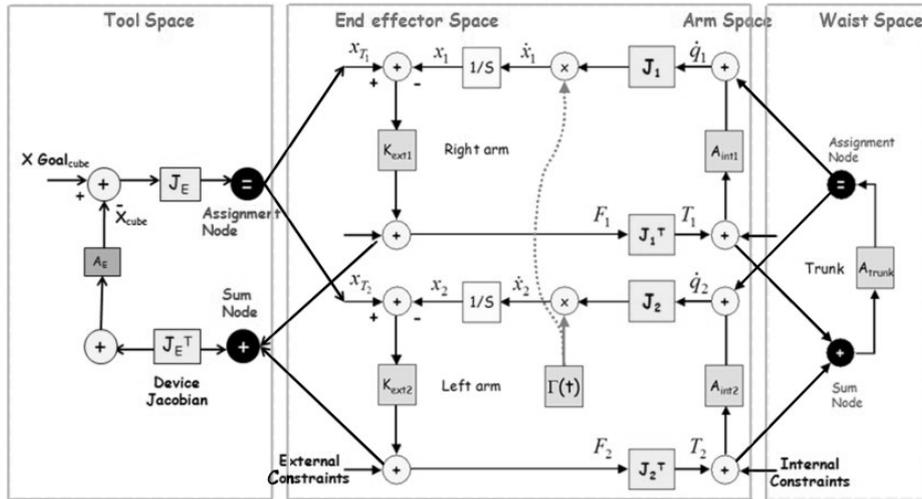


Figure 6. It shows the PMP network that coordinates the lifting phase of figure 5, with a force field applied to the object and propagated to the PMP sub-networks that correspond to the right arm, left arm, and trunk.

For the task of figure 6, figure 7 shows the time profiles of the extrinsic coordinates of the two end-effectors, the center of mass of the cube and the intrinsic coordinates of the two arms: all of them are coordinated by the same time base generator.

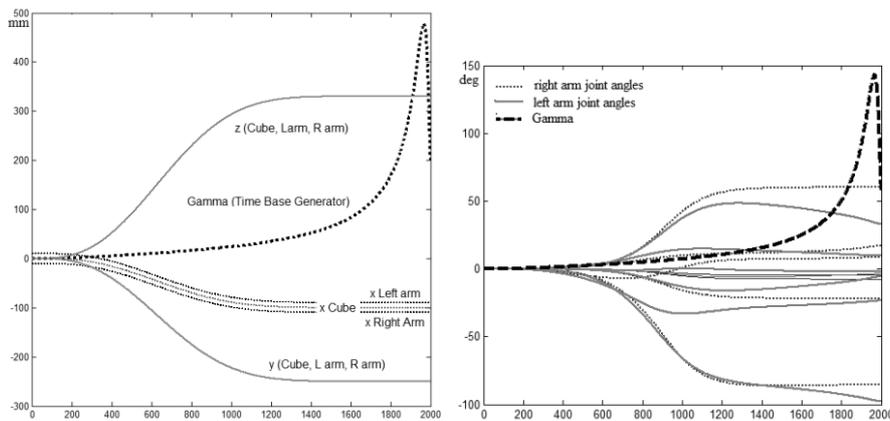


Figure 7. Kinematics of the lifting phase of the composite task depicted in fig. 6. Left panel: trajectory in the extrinsic space. Right panel: trajectory in the intrinsic space. Note that the same time base generator coordinates the smooth coordination of the joints and the end-effectors.

5. Discussion

In this chapter, we proposed a simple, distributed computational framework for representing and solving a range of ill-posed coordination problems arising in redundant humanoid platforms, by exploiting the solutions offered by the physics of passive motion and the concept of terminal attractors. Analogous to the coordination of

the movements of a wooden marionette by means of attached strings, the computational model operates by means of a relaxation to equilibrium of a network of task relevant parts of the body to the virtual pull induced by superimposed force fields, representing the goals and task relevant mixtures of constraints. The virtual force fields representing targets and constraints in different spaces are superimposed at runtime to yield a net force field that relaxes the internal model to a configuration: this solution is the best trade-off among the multiple set of constraints and is computed implicitly by the dynamics of the computational model.

We wish to emphasize that the force fields we are considering in this paper don't really describe the biomechanical forces at play during the execution of movements, but are computational metaphors that describe the complex dynamics of the internal computational engine. The internal model associated with the force-fields stores a whole family of geometrically possible solutions as a sort of a holographic memory, from which one solution is implicitly selected based on the nature of the task being executed and the pull of the field in the direction of the intended movement. Further, the proposed architecture is also endowed with nice computational properties like robustness, run time optimization, fast task adaptation, interference avoidance and local to global computation that makes it both biologically plausible and extremely useful in the control of complex robotic bodies.

The robustness of the computational machinery stems from several reasons:

1. No model inversion is needed as the system always operates by means of incremental well-posed, direct computations;
2. The dynamical system automatically stays away from singular configurations and relaxation to equilibrium in a conservative force field can never crash.

The system does not "search" for a solution, as it has access to a whole family of solutions stored in the form of a holographic memory, and "discovers" the one that is closest at hand. Even if the target is outside the reachable workspace, the robot nevertheless tries to approach the target as much as possible by fully extending the arm to a position that is at a minimum distance from the target. Hence, what we see in such cases is a *gentle degradation* of performance that characterizes humans in the same situations. Although there is no exact solution to the problem, the network "does its best".

The computational machinery is flexible because there are no pre-defined cost functions/optimization constraints in the model; hence there is a scope of operating online, facilitating runtime co-evolution of a plans and the corresponding control processes needed to achieve them. Multiple constraints can be concurrently imposed in a task-dependent fashion by simply switching on/off different task relevant force field generators.

From the perspective of local to global computing, at each instance of time, every element in the computational chain of figure 2 makes a (incremental) local decision regarding its contribution to the overall externally induced pull, based on its own compliance. All such local decisions contribute towards driving the system to a configuration that minimizes its global elastic potential energy. Similar to many connectionist models in the field of artificial neural networks, the mechanism to regularize and exploit redundancy by means of attaining configurations that minimize global potential energy essentially uses only local asynchronous interactions..

An interesting area of research closely related to the model proposed in this paper is the use of forward inverse internal models, now wide spread in the field of cognitive science. Existence of neural mechanisms that mimic input output characteristics and the

inverses of the motor apparatus are supported by several behavioral, neuropsychological and imaging data [26,27]. A forward model or an emulator is a computational mechanism that captures the forward or causal relationship between the inputs and outputs of a system. If we consider the arm as the target system, the forward model predicts the next state (position and velocity), given an initial state and motor command. The inverse model does the inverse of this task, i.e. it takes a goal end state as input and produces a sequence of motor commands necessary to achieve it.

It is quite easy to observe that the computational chain shown in figure 2 (and its extensions) essentially operate in both forward and inverse modes. In other words, the proposed computational architecture can be seen as a combination of a coupled pair of controllers: a forward motor controller that maps tentative trajectories in the joint space (intrinsic space) into the corresponding trajectories of the end-effector variables in the workspace (extrinsic space) and an inverse motor controller that maps desired trajectories of the end-effector into feasible trajectories in the joint space. Hence, a key element of the proposed architecture is that the same computational model is used to support mental simulations employed by the reasoning process and the actual delivery of motor commands during movement execution. The difference is that, in the latter case, the motor outflow interacts with the peripheral circuitry of the motor system (spinal circuitry and mechanical properties of the neuromuscular apparatus in the biological domain or its simplified equivalent in the robotic domain). This point of view is in agreement with the CODAM concept (Corollary Discharge of Attention Movement, [28]). We also note that unlike forward inverse models using supervised neural networks that operate by means of feed forward association between input and output patterns, the forward inverse model using the notion of passive motion paradigm operates by seeking stationary configurations of a non linear dynamical system.

From the perspective of reasoning, the model presented in this paper provides a unified computational machinery for both mental simulation of action employed by higher level reasoning processes and actual delivery of motor commands for real movement execution in the robot. In other words, one can reason about reaching without actually reaching and yet use the same neural/computational substrate to do so. The relaxation of the coupled forward inverse model pair provides a general solution for mentally simulating an action of reaching a target position taking into consideration a variety of geometric constraints (range of motion in the joint space, internal and external constraints in the workspace) as well as effort-related constraints (range of torque of the actuators, etc.). If the forward simulation is successful, the movement is executed; otherwise the residual "error" or measure of inconsistency can be used to trigger a higher level of reasoning regarding possible availability of a tool that could be used to get closer to the goal. Several studies from animal reasoning suggest that a number of species, including primates and crows, appear to be involved in some form of prospection and reasoning that involves using/making tools to achieve otherwise unreachable goals. Such experiments from animal reasoning form ideal scenarios for developing-validating computational architectures of cognitive behavior in humanoid robots.

From the perspective of linguistics, several proponents of the embodied interactionist theory of meaning have argued that the same neural substrate employed in mental simulation of action is also used in understanding actions of others and for grounding the meanings of verbs [29,30,31]. In other words, understanding a piece of language necessarily entails an internal mental simulation of the described scenario, by activating a subset of the neural structures that would be involved in perceiving the

percepts or performing the motor actions described. Future research in this direction with regards to computational modeling is expected to provide further insights into how language could be integrated with the motor knowledge in the cognitive system.

References

- [1] R.A. Brooks, The Cog project, *Journal of the Robotics Society of Japan* **15** (1997), 968-970.
- [2] C.G. Atkeson, J.G. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, M. Kawato, Using humanoid robots to study human behavior, *IEEE Intelligent Systems*, **15** (2000), 46-56.
- [3] M. Hirose, K. Ogawa Honda humanoid robots development, *Philos Transact A Math Phys Eng Sci*, **365** (2007) 11-19.
- [4] K. Nishiwaki, J. Kuffner, S. Kagami, M. Inaba, H. Inoue, The experimental humanoid robot H7: a research platform for autonomous behaviour, *Philos Transact A Math Phys Eng Sci*, **365** (2007) 79-107.
- [5] L. Natale, F. Orabona, G. Metta, G. Sandini, Sensorimotor coordination in a "baby" robot: learning about objects through grasping, *Prog Brain Res* **164** (2007) 403-424.
- [6] P. Morasso, Spatial control of arm movements, *Experimental Brain Research* **42** (1981), 223-227.
- [7] W. Abend, E. Bizzi, P. Morasso, Human arm trajectory formation, *Brain* **105** (1982), 331-348.
- [8] T. Flash, N. Hogan, The coordination of arm movements: an experimentally confirmed mathematical model, *J Neurosci* **5** (1985), 1688-703.
- [9] Y. Uno, M. Kawato, R. Suzuki, Formation and control of optimal trajectory in human multijoint arm movement. Minimum torque-change model, *Biol Cybern* **61** (1989), 89-101.
- [10] P. Morasso, V. Sanguineti, G. Spada A computational theory of targeting movements based on force fields and topology representing networks, *Neurocomputing* **15** (1997), 414-434.
- [11] E. Bizzi, F.A. Mussa-Ivaldi, S. Giszter, Computations underlying the execution of movement: a biological perspective, *Science* **253** (1991) 287-291.
- [12] R. Shadmehr, F.A. Mussa-Ivaldi, Adaptive representation of dynamics during learning of a motor task, *J Neurosci* **14** (1994), 3208-3224.
- [13] D.E. Whitney, Resolved Motion Rate Control of Manipulators and Human Prosthesis, *IEEE Transactions on Man-Machine Systems* **MMS-10** (1969), 47-53.
- [14] A. Liegeois, Automatic supervisory control of the configuration and behavior of multibody mechanisms, *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-7** (1977), 868-871.
- [15] J. Baillieul, Kinematic programming alternatives for redundant manipulators, *IEEE International Conference on Robotics and Automation* (1985), 722-728.
- [16] M. Šoeh, R. Lórencz, Solving inverse kinematics – a new approach to the extended Jacobian technique, *Acta Polytechnica* **45** (2005) 21-26.
- [17] F.A. Mussa Ivaldi, P. Morasso, R. Zaccaria, Kinematic networks: a distributed model for representing and regularizing motor redundancy, *Biological Cybernetics* **60**(1988), 1-16.
- [18] T. Tsuji, P. Morasso, K. Shigehashi, M. Kaneko, Motion planning for manipulators using artificial potential field approach that can adjust convergence time of generated arm trajectory, *Journal of the Robotics Society of Japan* **13** (1995), 285-290.
- [19] V. Mohan, P. Morasso, Towards reasoning and coordinating action in the mental space. *International Journal of Neural Systems* **17** (2007), 1-13.
- [20] D. Bullock, S. Grossberg, Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties. *Psychol Rev* **95** (1988) 49-90.
- [21] M. Hersch, A.G. Billard, Reaching with multi-referential dynamical systems. *Auton Robots* **25** (2008), 71-83.
- [22] A.J. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots. *Proced IEEE ICRA2002* (2002), 1398-1403.
- [23] M. Zak, Terminal attractors for addressable memory in neural networks, *Phys. Lett. A* **133** (1988), 218-222.
- [24] G. Metta, P. Fitzpatrick, L. Natale, YARP: Yet Another Robot Platform, *International Journal of Advanced Robotics Systems* **3** (2006), 43-48.
- [25] V. Tikhonoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, F. Nori, An Open-Source Simulator for Cognitive Robotics Research. *Cogprints*, (2008) article 6238.
- [26] D.M. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control, *Neural Networks* **11** (1998), 1317-1329.

- [27] E. Oztop, D.M. Wolpert, M. Kawato, Mental state inference using visual control parameters, *Cognitive Brain Research* **158** (2004), 480-503.
- [28] J.G. Taylor, The CODAM Model and Deficits of Consciousness, *Lecture notes in computer science*, **2774** (2003), Springer Berlin/Heidelberg.
- [28] V. Gallese, G. Lakoff, The brain's concepts: the role of the sensory-motor system in reason and language, *Cognitive Neuropsychology* **22** (2005), 455–479.
- [30] A.M. Glenberg, What memory is for, *Behaviour and Brain Sciences* **20** (1997), 1–19.
- [31] J. Feldman, S. Narayanan, Embodied meaning in a neural theory of language, *Brain and Language* **89** (2004), 385–392.